

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-082922

(43)Date of publication of application : 22.03.2002

(51)Int.Cl.

G06F 15/16

G06F 9/44

G06F 9/46

(21)Application number : 2001-180675

(71)Applicant : DIGITAL VISION LABORATORIES
CORP

(22)Date of filing : 05.03.1996

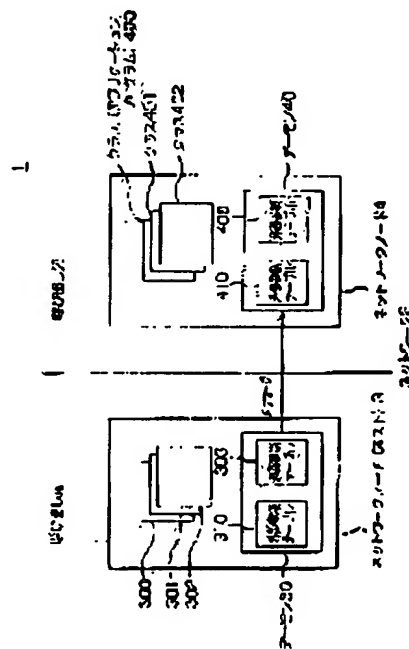
(72)Inventor : MAEKAWA HIROTOSHI
SAITO TAKAYUKI
CHIBA TETSUHISA

(54) PARALLEL DISTRIBUTION PROCESSING METHOD

(57)Abstract:

PROBLEM TO BE SOLVED: To provide a parallel distribution processing method capable of quickly and efficiently calling a function between systems.

SOLUTION: At the time of calling the function of a class 402 of a network node 4 by a class(module) 302 of a network node 3, the network node 3 acquires the function identification code(numeric value) of the function from an external reference table 310, and outputs the function identification code with an argument as a message to the network node 4. The network node 4 acquires the execution address of the function from an internal reference table 406 by using the inputted function identification code as a key, and executes the function.



LEGAL STATUS

[Date of request for examination] 14.06.2001

[Date of sending the examiner's decision of rejection] 11.10.2005

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

Searching PAJ

2/2 ページ

[Date of requesting appeal against examiner's
decision of rejection]

[Date of extinction of right]

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2002-82922

(P2002-82922A)

(43) 公開日 平成14年3月22日 (2002.3.22)

(51) Int.Cl.	識別記号	F I	テ-マ-ト* (参考)
G 0 6 F 15/16	6 2 0	G 0 6 F 15/16	6 2 0 S 5 B 0 4 5
9/44	5 3 0	9/44	5 3 0 M 5 B 0 9 8
9/46	3 6 0	9/46	3 6 0 B

審査請求 有 請求項の数 1 O L (全 13 頁)

(21) 出願番号 特願2001-180675(P2001-180675)
 (62) 分割の表示 特願平8-47833の分割
 (22) 出願日 平成8年3月5日 (1996.3.5)

(71) 出願人 396001380
 株式会社デジタル・ビジョン・ラボラトリーズ
 東京都港区赤坂七丁目3番37号
 (72) 発明者 前川 博俊
 東京都港区赤坂七丁目3番37号 株式会社
 デジタル・ビジョン・ラボラトリーズ内
 (72) 発明者 斎藤 隆之
 東京都港区赤坂七丁目3番37号 株式会社
 デジタル・ビジョン・ラボラトリーズ内
 (74) 代理人 100094053
 弁理士 佐藤 隆久

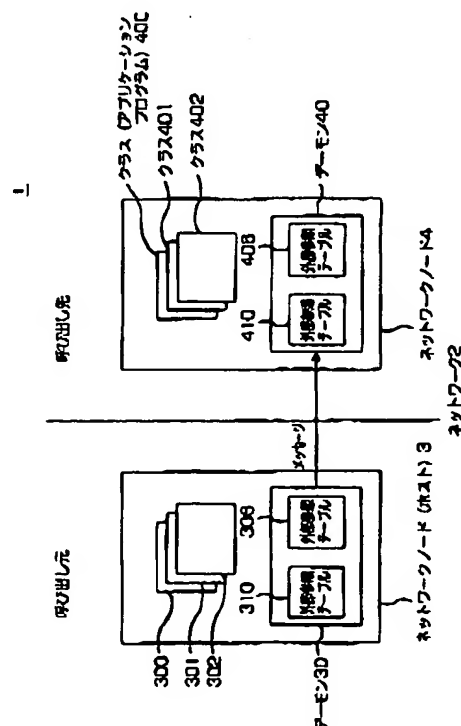
最終頁に続く

(54) 【発明の名称】 並列分散処理方法

(57) 【要約】

【課題】 システム相互間での機能呼び出しを高速かつ効率的に行うことができる並列分散処理方法を提供する。

【解決手段】 ネットワークノード3のクラス (モジュール) 302において、ネットワークノード4のクラス402の関数 (機能) の呼び出しを行うときに、ネットワークノード3は、外部参照テーブル310から当該関数の機能識別コード (数値) を得て、この機能識別コードが引数と共にメッセージとしてネットワークノード4に出力する。ネットワークノード4では、入力した機能識別コードをキーとして内部参照テーブル406から当該機能の実行アドレスを得て、当該機能を実行する。



(2)

特開2002-82922

【特許請求の範囲】

【請求項1】 ネットワークを介して接続された複数のネットワークノードの各々で、機能についての一連の手続きを記述したプログラムを動作させ、一のプログラムが、他のプログラムあるいは他のプログラムに記述された手続きによって規定された機能呼び出して実行する並列分散処理方法において、

前記複数のネットワークノード上の各々でデーモンが動作し、

前記デーモンは、

当該デーモンと同じネットワークノード上で動作する前記プログラムが規定する機能のうち他のプログラムから呼び出される機能について、当該機能の機能識別番号と当該機能の実行アドレスとの対応を示す内部参照テーブルと、

当該デーモンと同じネットワークノード上で動作する前記プログラムが呼び出しを行う他のプログラムの機能について、当該機能の機能識別子と当該機能の機能識別番号との対応を示す外部参照テーブルとを有し、

前記ネットワークノードは、前記並列分散処理の初期化時、あるいは、前記プログラムが追加された時に、前記機能の呼び出し関係に基づいて、前記内部参照テーブルおよび前記外部参照テーブルを生成あるいは更新し、

第1のネットワークノードが、前記機能の呼び出し元のプログラムに基づいて、呼び出しを行う機能の機能識別子をキーとして、当該呼び出し元のプログラムと同じネットワークノード上で動作する前記デーモンの前記外部参照テーブルから対応する機能識別番号を得て、当該機能識別番号を含むメッセージを、前記機能の呼び出し先のプログラムが動作する第2のネットワークノードに送り、

前記第2のネットワークノードが、前記呼び出し先のプログラムに基づいて、前記第1のネットワークノードから送られたメッセージに含まれる前記機能識別番号をキーとして、当該呼び出し先のプログラムと同じネットワークノード上で動作する前記デーモンの前記内部参照テーブルから当該呼び出された機能の実行アドレスを得て、当該実行アドレスに基づいて、当該呼び出された機能を実行し、

前記プログラムがオブジェクト指向プログラミングのクラスである場合に、当該クラスの内部変数を共有変数とし、当該共有変数へのアクセスを、当該共有変数を内包する前記クラスへの前記機能呼び出しによって実現する並列分散処理方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 本発明は、ネットワークワイドなマルチメディア並行処理を行う並列分散処理方法に関する。

【0002】

【従来の技術】 近年、マルチメディア情報を利用および流通させる情報提供利用サービスアプリケーションであるISM(Information Super-Market)の開発が活発に行われている。ISMでは、機能モジュール(クラス)間の対話性およびモジュール間構成の柔軟性を高めるために、ネットワークワイドなマルチメディア並行処理を効率的に行うアプリケーション実行環境が求められる。これに応えようとするのが、並列分散実行支援システムである。従来の並列分散実行支援システムでは、モジュール間の機能呼び出しは、文字列で示されるモジュール名と機能名とを用いて行う。ここで、機能呼び出しとは、あるモジュールで行われる処理において、他のモジュールにおいて備えてある機能呼び出して用いることをいう。

【0003】 並列分散支援システムの代表的なものに、CORBA(Common Object Request Broker Architecture) - ORB(Object Request Broker)がある。このCORBA-ORBでは、あるモジュールが他のモジュールの機能呼び出すにあたって、文字列で示される機能名と引数が呼び出し元モジュールから呼び出し先モジュールに転送される。呼び出し先モジュールには、文字列で示される機能名と当該機能を実行するプログラムの実行アドレスとの対応表が備えてあり、呼び出し先モジュールは、その対応表を検索することによって実行アドレスを取得し、呼び出された機能を実行する。

【0004】

【発明が解決しようとする課題】 しかしながら、前述した従来のCORBA-ORBでは、機能の呼び出し先において、文字列で示される機能名をキーとして、テーブルを参照して実行アドレスを特定することから、その実行部を特定するコストが大きく、機能呼び出しに伴う処理の効率が悪いという問題がある。

【0005】 また、CORBA-ORBでは、モジュールの識別名の規約とその管理方法については規定されていない、すなわち、システム全体での名前空間管理は行われていない。従って、ユーザは、モジュール相互間で名前の衝突などが生じないように注意して、モジュールの識別名などを決定しなければならない、ユーザの負担が大きいう問題がある。

【0006】 また、CORBA-ORBでは、機能呼び出しに伴う引数の形式(フォーマット)は、コンパイル時に別途定義しておく必要がある。このとき、定義できる引数のデータ構造は構造体および配列などの単純なものに限られ、リスト構造などの複雑なデータ構造は用いることができないという問題がある。

【0007】 また、CORBA-ORBでは、呼び出し先モジュールは予め存在(登録)している必要がある。また、基本的に、呼び出し機能および引数の形式をコンパイル時に定義しておく必要があり、任意の呼び出し先のモジュールを動的にダウンロード(追加登録)して使

(3)

特開2002-82922

用しようとする、作業が煩雑になるという問題がある。また、CORBA-ORBでは、機能呼び出しを行う際の同期方式が異なると、異なるプログラミングインターフェース(API)を用いる必要があり、プログラムの記述が複雑になるという問題がある。

【0008】本発明は、上述した従来技術に鑑みてなされ、機能呼び出しを高速かつ効率的に行える並列分散処理方法を提供することを目的とする。また、本発明は、ユーザによるモジュールの識別名の管理負担を軽減できる並列分散処理方法を提供することを目的とする。また、本発明は、機能呼び出しにおいて、多様な形式の引数を用いることができる並列分散処理方法を提供することを目的とする。また、本発明は、煩雑な作業を経ることなく、任意の呼び出し先のモジュールを動的にダウンロード(追加登録)して使用できる並列分散処理方法を提供することを目的とする。

【0009】

【課題を解決するための手段】上述した従来技術の問題点を解決し、上述した目的を達成するために、本発明の並列分散処理方法は、ネットワークを介して接続された複数のネットワークノードの各々で、機能についての一連の手続きを記述したプログラムを動作させ、一のプログラムが、他のプログラムあるいは他のプログラムに記述された手続きによって規定された機能を読み出して実行する並列分散処理方法において、前記複数のネットワークノード上の各々でデーモンが動作し、前記デーモンは、当該デーモンと同じネットワークノード上で動作する前記プログラムが規定する機能のうち他のプログラムから呼び出される機能について、当該機能の機能識別番号と当該機能の実行アドレスとの対応を示す内部参照テーブルと、当該デーモンと同じネットワークノード上で動作する前記プログラムが呼び出しを行う他のプログラムの機能について、当該機能の機能識別子と当該機能の機能識別番号との対応を示す外部参照テーブルとを有し、前記ネットワークノードは、前記並列分散処理方法の初期化時、あるいは、前記プログラムが追加された時に、前記機能の呼び出し関係に基づいて、前記内部参照テーブルおよび前記外部参照テーブルを生成あるいは更新し、第1のネットワークノードが、前記機能の呼び出し元のプログラムに基づいて、呼び出しを行う機能の機能識別子をキーとして、当該呼び出し元のプログラムと同じネットワークノード上で動作する前記デーモンの前記外部参照テーブルから対応する機能識別番号を得て、当該機能識別番号を含むメッセージを、前記機能の呼び出し先のプログラムが動作する第2のネットワークノードに送り、前記第2のネットワークノードが、前記呼び出し先のプログラムに基づいて、前記第1のネットワークノードから送られたメッセージに含まれる前記機能識別番号をキーとして、当該呼び出し先のプログラムと同じネットワーク上で動作する前記デーモンの前記内部参

照テーブルから当該呼び出された機能の実行アドレスを得て、当該実行アドレスに基づいて、当該呼び出された機能を実行し、前記プログラムがオブジェクト指向プログラミングのクラスである場合に、当該クラスの内部変数を共有変数とし、当該共有変数へのアクセスを、当該共有変数を内包する前記クラスへの前記機能呼び出しによって実現する。

【0010】また、後述する実施形態には、以下の発明が開示されている。第1の発明は、ネットワークを介して接続された複数のネットワークノードの各々で、機能についての一連の手続きを記述したプログラムを動作させ、一のプログラムが、他のプログラムあるいは他のプログラムに記述された手続きによって規定された機能を読み出して実行する並列分散処理方法において、前記複数のネットワークノード上の各々でデーモンが動作し、前記デーモンは、当該デーモンと同じネットワークノード上で動作する前記プログラムが規定する機能のうち他のプログラムから呼び出される機能について、当該機能の機能識別番号と当該機能の実行アドレスとの対応を示す内部参照テーブルと、当該デーモンと同じネットワークノード上で動作する前記プログラムが呼び出しを行う他のプログラムの機能について、当該機能の機能識別子と当該機能の機能識別番号との対応を示す外部参照テーブルとを有し、前記ネットワークノードは、前記並列分散処理の初期化時、あるいは、前記プログラムが追加された時に、前記機能の呼び出し関係に基づいて、前記内部参照テーブルおよび前記外部参照テーブルを生成あるいは更新し、第1のネットワークノードが、前記機能の呼び出し元のプログラムに基づいて、呼び出しを行う機能の機能識別子をキーとして、当該呼び出し元のプログラムと同じネットワークノード上で動作する前記デーモンの前記外部参照テーブルから対応する機能識別番号を得て、当該機能識別番号を含むメッセージを、前記機能の呼び出し先のプログラムが動作する第2のネットワークノードに送り、前記第2のネットワークノードが、前記呼び出し先のプログラムに基づいて、前記第1のネットワークノードから送られたメッセージに含まれる前記機能識別番号をキーとして、当該呼び出し先のプログラムと同じネットワーク上で動作する前記デーモンの前記内部参照テーブルから当該呼び出された機能の実行アドレスを得て、当該実行アドレスに基づいて、当該呼び出された機能を実行し、前記プログラムがオブジェクト指向プログラミングのクラスである場合に、当該クラスの内部変数を共有変数とし、当該共有変数へのアクセスを、当該共有変数を内包する前記クラスへの前記機能呼び出しによって実現する並列分散処理方法。

【0011】第2の発明は、前記機能識別子は、前記機能識別番号が格納されているアドレスを示す第1の発明の並列分散処理方法。第3の発明は、前記機能識別子は、前記呼び出し元のプログラムにおいてのみ有効な番

(4)

特開2002-82922

号を示す第1の発明に記載の並列分散処理方法。第4の発明は、前記機能識別子は、文字を含んで表現される機能名を示す第1の発明に記載の並列分散処理方法。

【0012】第5の発明は、前記第1のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記第2のネットワークノードに送る引数を、当該引数のデータ構造を論理形式表現した引数に変換し、当該変換後の引数を前記第2のネットワークノードに出力し、前記第2のネットワークノードは、前記第1のネットワークノードから入力した引数のデータ構造を前記呼び出し元における当該引数のデータ構造と同じデータ構造に変換した後に、当該引数を使用する第1の発明から第4の発明のいずれかに記載の並列分散処理方法。

【0013】第6の発明は、前記第1のネットワークノードは、前記呼び出し元のプログラムに基づいて、呼び出し先のプログラムに前記機能識別番号を出力した後、前記第2のネットワークノードから当該機能の実行結果を入力するまで、その処理を停止する第1の発明から第5の発明のいずれかに記載の並列分散処理方法。

【0014】第7の発明は、前記第1のネットワークノードは、前記呼び出し元のプログラムに基づいて、呼び出し先のプログラムに前記機能識別番号を出力した後、当該機能の実行結果が必要となったときに、当該機能の実行結果をまだ入力していない場合に、その処理を停止する第1の発明から第5の発明のいずれかに記載の並列分散処理方法。

【0015】第8の発明は、前記第1のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記第2のネットワークノードに前記機能識別番号を出力した後、当該機能の実行結果を入力しない第1の発明から第5の発明のいずれかに記載の並列分散処理方法。

【0016】第9の発明は、前記第1のネットワークノードは、前記呼び出し元のプログラムに基づいて、呼び出し先のプログラムに前記機能識別番号を出力した後、所定時間経過しても当該機能の実行結果を入力しない場合に、エラー処理プログラムを起動する第1の発明から第7の発明のいずれかに記載の並列分散処理方法。

【0017】第10の発明は、前記第1のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記メッセージに送信時刻を含め、前記第2のネットワークノードは、前記送信時刻に基づいて、呼び出された機能の実行時刻を決定する第1～第9の発明のいずれかに記載の並列分散処理方法。

【0018】第11の発明は、前記第1のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記第2のネットワークノードに複数のメッセージを送る場合に、これらのメッセージに対応する複数の機能呼び出しを、予め決められた優先順位あるいはメッセージにおいて指定された優先順位に従って順に行う第1～10の発明のいずれかに記載の並列分散処理方法。

【0019】第12の発明は、前記第1のネットワークノードおよび前記第2のネットワークノードは、新たなプログラムが登録されたときに、前記デーモンに基づいて、そのプログラムが呼び出し先および呼び出し元となる機能について解析し、その解析結果に基づいて、前記内部参照テーブルおよび外部参照テーブルを更新する第1～11の発明のいずれかに記載の並列分散処理方法。

【0020】第13の発明は、前記プログラムおよび前記機能は、それぞれオブジェクト指向プログラミングにおけるクラスおよびメソッドである第1～12の発明のいずれかに記載の並列分散処理方法。

【0021】第14の発明は、前記プログラムがクラスであり、当該クラスの内部変数を共有変数として用いる場合に、前記共有変数の読み取りまたは変更を、当該クラスのメソッドで規定する第1～13の発明のいずれかに記載の並列分散処理方法。

【0022】第15の発明は、ネットワークを介して接続された複数のネットワークノードを有し、前記複数のネットワークノードの各々で、機能についての一連の手続きを記述したプログラムを動作させ、一のプログラムが、他のプログラムあるいは他のプログラムに記述された手続きによって規定された機能呼び出しを実行する並列分散処理システムにおいて、前記複数のネットワークノード上の各々でデーモンが動作し、前記デモードは、当該デモンと同じネットワークノード上で動作する前記プログラムが規定する機能のうち他のプログラムから呼び出される機能について、当該機能の機能識別番号と当該機能の実行アドレスとの対応を示す内部参照テーブルと、当該デモンと同じネットワークノード上で動作する前記プログラムが呼び出しを行う他のプログラムの機能について、当該機能の機能識別子と当該機能の機能識別番号との対応を示す外部参照テーブルとを有し、前記ネットワークノードは、前記並列分散システムの初期化時、あるいは、前記プログラムが追加された時に、前記機能の呼び出し関係に基づいて、前記内部参照テーブルおよび前記外部参照テーブルを生成あるいは更新し、第1のネットワークノードが、前記機能の呼び出し元のプログラムに基づいて、呼び出しを行う機能の機能識別子をキーとして、当該呼び出し元のプログラムと同じネットワークノード上で動作する前記デーモンの前記外部参照テーブルから対応する機能識別番号を得て、当該機能識別番号を含むメッセージを、前記機能の呼び出し先のプログラムが動作する第2のネットワークノードに送り、前記第2のネットワークノードが、前記呼び出し先のプログラムに基づいて、前記第1のネットワークノードから送られたメッセージに含まれる前記機能識別番号をキーとして、当該呼び出し先のプログラムと同じネットワーク上で動作する前記デーモンの前記内部参照テーブルから当該呼び出された機能の実行アドレスを

(5)

特開2002-82922

得て、当該実行アドレスに基づいて、当該呼び出された機能を実行する並列分散処理システム。

【0023】第16の発明は、前記機能識別子は、前記機能識別番号が格納されているアドレスを示す第15の発明に記載の並列分散処理システム。

【0024】第17の発明は、前記機能識別子は、前記呼び出し元のプログラムにおいてのみ有効な番号を示す第15の発明に記載の並列分散処理システム。

【0025】第18の発明は、前記機能識別子は、文字を含んで表現される機能名を示す第15の発明に記載の並列分散処理システム。第19の発明は、前記第1のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記第2のネットワークノードに送る引数を、当該引数のデータ構造を論理形式表現した引数に変換し、当該変換後の引数を前記第2のネットワークノードに出力し、前記第2のネットワークノードは、前記第1のネットワークノードから入力した引数のデータ構造を前記呼び出し元における当該引数のデータ構造と同じデータ構造に変換した後に、当該引数を使用する第15～18の発明のいずれかに記載の並列分散処理システム。第20の発明は、前記第1のネットワークノードは、前記呼び出し元のプログラムに基づいて、呼び出し先のプログラムに前記機能識別番号を出力した後、前記第2のネットワークノードから当該機能の実行結果を入力するまで、その処理を停止する第15～19の発明のいずれかに記載の並列分散処理システム。第21の発明は、前記第1のネットワークノードは、前記呼び出し元のプログラムに基づいて、呼び出し先のプログラムに前記機能識別番号を出力した後、当該機能の実行結果が必要となったときに、当該機能の実行結果をまだ入力していない場合に、その処理を停止する第15～19の発明のいずれかに記載の並列分散処理システム。第22の発明は、前記第1のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記第2のネットワークノードに前記機能識別番号を出力した後、当該機能の実行結果を入力しない第15～19の発明のいずれかに記載の並列分散処理システム。第23の発明は、前記第1のネットワークノードは、前記呼び出し元のプログラムに基づいて、呼び出し先のプログラムに前記機能識別番号を出力した後、所定時間経過しても当該機能の実行結果を入力しない場合に、エラー処理プログラムを起動する第15～21の発明のいずれかに記載の並列分散処理システム。第24の発明は、前記第1のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記メッセージに送信時刻を含め、前記第2のネットワークノードは、前記送信時刻に基づいて、呼び出された機能の実行時刻を決定する第15～23の発明のいずれかに記載の並列分散処理システム。第25の発明は、前記第1のネットワークノードは、前記呼び出し元のプログラムに基づいて、前記第2のネットワークノードに複数のメッ

セージを送る場合に、これらのメッセージに対応する複数の機能呼び出しを、予め決められた優先順位あるいはメッセージにおいて指定された優先順位に従って順に行う第15～24の発明のいずれかに記載の並列分散処理システム。第26の発明は、前記第1のネットワークノードおよび前記第2のネットワークノードは、新たなプログラムが登録されたときに、前記デーモンに基づいて、そのプログラムが呼び出し先および呼び出し元となる機能について解析し、その解析結果に基づいて、前記内部参照テーブルおよび外部参照テーブルを更新する第15～25の発明のいずれかに記載の並列分散処理システム。第27の発明は、前記プログラムおよび前記機能は、それぞれオブジェクト指向プログラミングにおけるクラスおよびメソッドである第15～26の発明のいずれかに記載の並列分散処理システム。第28の発明は、前記プログラムがクラスであり、当該クラスの内部変数を共有変数として用いる場合に、前記共有変数の読み取りまたは変更を、当該クラスのメソッドで規定する第15～27の発明のいずれかに記載の並列分散処理システム。

【0026】

【発明の実施の形態】以下、本発明の実施形態に係わる並列分散実行支援システムについて説明する。本実施形態に係わる並列分散実行支援システムは、例えば、ISMなどの実行環境を支援し、機能モジュール間の対話性およびモジュール間構成の柔軟性を高め、ネットワークワイドなマルチメディア並行処理を可能とするものである。

【0027】図1は、本実施形態に係わる並列分散実行支援システム1の概念図である。図1に示すように、並列分散実行支援システム1は、ネットワークノード3、4がネットワーク2を介して通信を行い、一定の処理を行う。ネットワークノード3、4としては、例えば、ホストコンピュータなどが用いられる。ネットワークノード3では、デーモン30およびクラス（モジュール）300、301、302などのプログラムが動作する。ここで、クラスとは、具体的には、例えば、アプリケーションプログラムをいう。

【0028】デーモン30は、後述するように、内部参照テーブル306および外部参照テーブル310を有する。内部参照テーブル306は、機能識別コードとその機能が記憶されたアドレスとの対応を示すテーブルである。内部参照テーブル306は、ネットワークノード4からネットワークノード3に機能呼び出しを行うときに、ネットワークノード4から入力したメッセージに含まれる機能識別コードから、呼び出された機能のクラス300～302におけるアドレスを得るために用いられる。

【0029】外部参照テーブル310は、機能識別子としての機能識別コードが記憶されたアドレスと機能識別

(6)

特開2002-82922

コードとの対応を示すテーブルである。外部参照テーブル310は、ネットワークノード3からネットワークノード4に機能呼び出しを行うときに、クラス300~302において呼び出されたアドレスから、その機能の機能識別コードを得るために用いられる。

【0030】また、ネットワークノード4では、デーモン40およびクラス（モジュール）400、401、402などのプログラムが動作する。ネットワークノード4のクラス400~402、内部参照テーブル406および外部参照テーブル410は、その意味および機能は、ネットワークノード3のクラス300~302、内部参照テーブル306および外部参照テーブル310と同じである。

【0031】図1に示す並列分散実行支援システム1では、クラスの配置、すなわち、どのクラスをどのネットワークノードあるいはプロセッサに配置したかは、原則として、システムのセットアップ時に一回のみ解析（指定）される。従って、システムの実行時において、各クラスは、他のクラスの機能を読み出す際に読み出し先のクラスの所在は既に知っている。尚、クラスの配置は、システムの実行時に指定されることもある。

【0032】尚、ネットワーク2には、ネットワークノード3、4に加えて他のネットワークノードが接続されていてもよい。

【0033】以下、オブジェクト指向の計算におけるメソッド呼び出しの処理、すなわち、ネットワークノード3でクラス302の処理を行っているときに、ネットワークノード4のクラス402に備えられた機能、具体的には関数を読み出す場合を例示して、並列分散実行支援システム1における処理について説明する。オブジェクト指向の計算におけるメソッド呼び出しの処理では、クラス内のメソッドが例えば関数などの機能である。

【0034】

内部参照テーブルおよび外部参照テーブルの作成

並列分散実行支援システム1では、例えば、図3に示すネットワークノード3においてクラス302が新たに追加（登録）され、クラス302が、図2に示すネットワークノード4のクラス402の機能を読み出している場合には、以下に示すように、図3に示すデーモン30の外部参照テーブル310および図2に示すデーモン40の内部参照テーブル406を作成あるいは更新する。

【0035】まず、内部参照テーブルの作成について説明する。ネットワークノード3、4は、例えば、システムのセットアップ時に、それらのクラスで規定された関数について、図2、図3に示す内部参照テーブル406、306を作成する。ここで、内部参照テーブル306は、ローカルクラステーブル303、ローカルメンバ関数テーブル304およびローカルインスタンステーブル305で構成される。

【0036】また、内部参照テーブル406は、ローカ

ルクラステーブル403、ローカルメンバ関数テーブル404およびローカルインスタンステーブル405で構成される。ここで、ローカルメンバ関数テーブル304、404は、関数の「関数ID」と、関数「Func」の実行アドレスである「関数番地」との対応を示している。尚、ネットワークノード3、4を含むネットワークノード相互間での関数の呼び出し関係は、システムのセットアップ時およびクラス302が新たに追加されたときに予め解析されており、外部参照テーブル310および内部参照テーブル406を作成する段階において、関数「Func」およびクラス402にはそれぞれ「関数ID（機能識別コード）」および「クラスID」が自動的に割り当てられる。

【0037】次に、外部参照テーブルの作成について説明する。例えば、図1に示すクラス302がクラス402に備えてある関数名「Func」という機能を読み出している場合に、図3に示すように、ネットワークノード3は、クラス302のディスパッチャ311から、クラス402および関数名「Func」を用いて、ネットワークノード4に問い合わせ312を出力する。

【0038】ネットワークノード4は、この問い合わせ312を受けると、内部参照テーブル406を参照して、クラス402および関数「Func」に割り付けた「クラスID」および「関数ID」を図3に示す回答メッセージ315に含めて、図1に示すネットワーク2を介してネットワークノード3に出力する。回答メッセージ315は、ネットワークノード3において、メッセージキュー（待ち行列）312、313を介してディスパッチャ311に出力される。

【0039】次に、ディスパッチャ311は、登録処理を行い、回答メッセージ314に含まれる「クラスID」および「関数ID」を用いて、リモートクラステーブル307およびリモートメンバ関数テーブル308を作成する。ここで、リモートクラステーブル307、リモートメンバ関数テーブル308およびリモートインスタンステーブル309によって、外部参照テーブル310が構成される。

【0040】内部参照テーブルおよび外部参照テーブルの作成は、ネットワークノード3とネットワークノード4とがネットワーク2を介して接続されたときに初期化過程として行われたり、処理の実行時に、ネットワークノード3、4に新たなクラス（モジュール）が動的にダウンロード（追加登録）されたときに、一度のみ自動的に行われる。

【0041】機能呼び出し処理

以下、前述したように作成された外部参照テーブル310および内部参照テーブル406を用いて、ネットワークノード3からネットワークノード4に対して機能呼び出しを行う場合について図2~6を参照しながら説明する。ここでは、呼び出し元のクラス（オブジェクト）

(7)

特開2002-82922

が、機能呼び出しを行った後に、呼び出し先のクラスから当該機能の実行結果を入力するまで、その処理を停止する、いわゆる完全同期メッセージ送信の場合について説明する。

【0042】例えば、図4に示すネットワークノード3のクラス302において処理が行われているときに、ネットワークノード4に備えてあるクラス402の関数「Func」という機能呼び出すと（図6（A）に示す「A1」）、クラス302のディスパッチャ311において、メッセージに含まれる引数のデータ構造が論理形式のデータ表現に変換される（図6（A）に示す「B1」）。当該変換は、変換プログラムを用いて、クラス302での内部表現データのデータ構造を解析し、論理形式のデータ表現に変換して行われる。このとき、論理形式のデータ表現は、単純なデータを組み合わせることで、リスト構造などの複雑なデータ構造を表現できる記述構文を採用している。そのため、引数として、多様なデータ構造を用いることができる。そのため、機能呼び出しの対象とできる関数の種類の範囲を拡大できる。

【0043】次に、ディスパッチャ311において、関数「Func」およびクラス402のアドレスが、それぞれリモートメンバ関数テーブル308およびリモートインスタンステーブル309に出力される。これによって、機能識別コードの参照が行われ、これらのアドレスに対応した「関数ID」および「クラスID」が検索される（図6（A）に示す「B2」）。この検索された「関数ID」および「クラスID」は、ディスパッチャ311において、送信キュー313に登録される（図6（A）に示す「B3」）。

【0044】次に、送信キュー313に登録された「関数ID」および「クラスID」が、デーモン30の送信キュー314に登録される（図6（A）に示す「C1」）。デーモン30において、メッセージ320のメッセージヘッダが「関数ID」および「クラスID」を用いて作成され、このメッセージヘッダを含むメッセージが、図5に示すネットワークノード4のデーモン40の受信キュー414に、ネットワークを介して送信される（図6（A）に示す「C2」）。ネットワークノード3は、ネットワークノード4から関数「Func」の処理結果を入力するまで、実行中のプログラムの処理を停止する。このように、メッセージキュー314、414を用いることで、メッセージは、キューに蓄えられた後に、該当関数（機能）によって処理される。そのため、これらのキューによって、ある特定のクラスに対しての機能呼び出し要求が競合しても、呼び出し要求が失われることなく、順番にまたは優先順位に従って処理される。

【0045】ネットワークノード4のデーモン40は、ネットワークノード3からメッセージを受信すると（図6（B）に示す「D1」）、そのメッセージを図5に示

す受信キュー414に登録する（図6（B）に示す「D2」）。次に、デーモン40からディスパッチャ411にシグナルが出力される（図6（B）に示す「D3」、「E1」）。

【0046】次に、ディスパッチャ411は、「クラスID」をキーとして、図5に示すローカルインスタンステーブル405を参照し、所定のインスタンス呼び出す（図6（B）に示す「E2」）。ディスパッチャ411は、所定のインスタンスが呼び出し可能な状態になるまで、次の処理を実行する（図6（B）に示す「E3」）。このとき、メッセージキュー414に設けられた通信マネージャは、メッセージ320に含まれる「クラスID」をキーとして、ローカルインスタンステーブル405が参照され、クラス402のインスタンスの番地を得る。これによって、クラス402のインスタンスが決定される。ここで、インスタンスとは、クラスの処理を実際に行うときの状態を示す実体をいう。

【0047】尚、クラス402におけるインスタンスの生成は、図5に示すように、ローカルクラステーブル403においてクラス402と対応して記憶された「インスタンス生成関数の番地」を参照し、この「インスタンス生成関数の番地」にあるインスタンス生成関数421を実行し、これによって生成したインスタンスをローカルインスタンステーブル405に登録して行われる。

【0048】次に、ディスパッチャ411は、メッセージのデータ構造を元のクラス302で用いられているデータ構造と同じものに変換する（図6（B）に示す「E4」）。

【0049】次に、受信したメッセージ320に含まれる「関数ID」および「クラスID」に基づいて、ローカルメンバ関数テーブル404を用いて、呼び出された機能に関する関数「Func」の「関数番地」を得る。このとき、「関数ID」および「クラスID」は数値で示されているため、その照合は高速に行われる。

【0050】ネットワークノード4では、ディスパッチャ411において得られた関数「Func」の「関数番地」、通信マネージャによって決定されたインスタンスおよびディスパッチャ411において変換されて得られた内部表現データ420を用いて、クラス402における処理が行われる（図6（B）に示す「F」）。この処理結果は、返却値として、図6（B）に示す「E5」、「D4」、「D5」および図6（A）に示す「C3」、「C4」、「C5」、「B4」、「B5」および「A2」の処理を行うことで、図3に示す呼び出し元のネットワークノード3のクラス302の処理に送られる。これによって、機能呼び出し処理が完了する。

【0051】ここで、「E5」は返却値のデータ表現の変換、「D4」は送信キューへの返却値の登録、「D5」はメッセージの送信、「C3」はメッセージの受信、「C4」は受信キューへの登録、「C5」はクラス

(8)

特開2002-82922

302へのシグナル送信、「B4」はシグナル受信、「B5」は返却値のデータ表現の変換および「A2」は次の処理の継続実行を意味する。ネットワークノード3は、ネットワークノード4から関数「Func」の処理結果を入力すると、停止していたプログラムの処理を再開する。

【0052】以上説明したように、並列分散実行支援システム1によれば、関数などの機能の呼び出しは、クラス相互間での機能呼び出し関係および呼び出される機能についての数値で示される機能識別コードなどの関係を示した内部参照テーブルおよび外部参照テーブルを用いて行われるため、呼び出された機能の実行部のアドレスを高速に特定することができる。また、並列分散実行支援システム1によれば、機能呼び出しが行われるクラスおよび関数のアドレスについて、システムが自動的にユニークな機能識別コードを付与する。そのため、ユーザ（プログラマ）はクラスおよび関数のアドレスが競合することを回避するために多大な労力を費やす必要はなく、ユーザの負担が軽減される。

【0053】また、並列分散実行支援システム1によれば、クラスの追加に応じて内部参照テーブルおよび外部参照テーブルを動的に作成することから、必要なクラス（モジュール）を動的に指定したネットワークノードにダウンロードすることが可能となる。また、並列分散実行支援システム1によれば、機能呼び出しの引数と処理結果（返り値）の形式（フォーマット）をコンパイル時に定義しておく必要がないため、機能呼び出しの柔軟性を高めることができる。

【0054】ところで、プロセッサなどのシステムの資源を効率よく使用するために、ユーザ（プログラマ）が意識しないレベルでプログラムを細かい並行分散実行単位に自動的に分割するプログラミングシステムは存在する。これに対して、並列分散実行支援システム1は、ユーザがアプリケーションプログラムを大きな機能単位で並行分散モジュールに分割し、その並行分散モジュールの配置を明示的に指定できるものであり、大きなアプリケーションシステムの構築を可能とするものである。

【0055】次に、ネットワークノード3からネットワークノード4に対して機能呼び出しを行う場合についての他の例を説明する。ここでは、呼び出し元のクラス（オブジェクト）が、機能呼び出しを行った後に、呼び出し先のクラスから当該機能の実行結果を入力しない、いわゆる非同期メッセージ送信の場合について説明する。

【0056】図7は、非同期メッセージ送信の場合のタイミングチャートである。すなわち、図4に示すネットワークノード3のクラス302において処理が行われているときに、ネットワークノード4に備えてあるクラス402の関数「Func」を非同期のメッセージ送信で呼び出すと（図7（A）に示す「G1」）、ネットワー

クノード3のクラス302、ディスパッチャ311およびデーモン30において、図7（A）に示す「H1」、「H2」、「H3」、「I1」、「I2」および「G2」の処理が実行される。

【0057】図7（A）に示す「H1」、「H2」、「H3」、「I1」および「I2」の処理は、前述した図6（A）に示す「B1」、「B2」、「B3」、「C1」および「C2」の処理に対応する。ただし、図7に示す例では、クラス302は、呼び出しを行ったネットワークノード4のクラス402の機能の処理結果を待つことはせず、処理「I2」の後に、次の処理「G2」を実行する。

【0058】一方、ネットワークノード4では、デーモン40、ディスパッチャ411およびクラス402において、図7（B）に示す「J1」、「J2」、「J3」、「K1」、「K2」、「K3」、「K4」および「L」の処理を実行する。ここで、図7（B）に示す「J1」、「J2」、「J3」、「K1」、「K2」、「K3」、「K4」および「L」の処理は、図6（B）に示す「D1」、「D2」、「D3」、「E1」、「E2」、「E3」、「E4」および「F」の処理と、基本的には同じであるが、「F」の処理結果は、ネットワークノード3には送信されない。

【0059】次に、上述した本実施形態に係わる並列分散実行支援システム1の概念についてまとめる。図8に示すように、ネットワークノード98は、オブジェクト101aにおけるメソッド101bにおいて、ネットワークノード99のメソッド105aのメソッド呼び出しを行っている。このメソッド呼び出しの処理において、メソッド本体106のIDを示すメソッド101bのメソッドIDがメッセージに含められる。また、引数101cは、そのデータ構造が論理形式に変換された引数101c1となった後にメッセージに含められる。また、オブジェクト101aからの呼び出しキーに基づいて、外部参照表102が参照され、オブジェクトIDがメッセージに含められる。メッセージは、ネットワークノード98からネットワークノード99に送信される。

【0060】ネットワークノード99では、ネットワークノード98から送信されたオブジェクトIDをキーとして、内部参照表103を用いて、呼び出し対象となっているメソッド105aのメソッド本体106が実行される。このとき、論理変換された引数101c1が、オブジェクト101aにおける引数のデータ構造と同じデータ構造に変換され、この変換された引数105bを用いて、メソッド本体106の処理が実行される。

【0061】本発明は、上述した実施形態には限定されない。例えば、上述した実施形態では、本発明をオブジェクト指向プログラミングにおけるメソッドの呼び出しに適用した場合について例示したが、本発明は、例えば、リモート関数呼び出しにも適用できる。リモート関

(9)

特開2002-82922

数呼び出しは、リモートのアプリケーションなどのモジュールに存在する関数（機能）を呼び出すものである。

【0062】また、本発明は、あるクラスの内部変数を分散モジュール（クラス）群間の共有変数（大域変数）として使用する場合にも適用できる。この場合には、共有変数となる変数を内包するクラスの機能として、共有変数の値の読み取り、変更などの機能をプログラムする。このとき、共有変数へのアクセスは、全て、その変数を内包するクラスへの機能呼び出しによって行われるので、自然に、その共有変数へのアクセスの排他制御が実現される。すなわち、共有変数は、インスタンスの中に抱え込まれており、当該共有変数を読み出しおよび書き換えて操作するには、当該インスタンスの機能を呼び出す必要がある。これによって、インスタンスが共有変数そのものとして振る舞う。このとき、共有変数をオブジェクト（クラス）とし、その共有変数についての処理をメソッド（関数）とする。

【0063】また、上述した実施形態では、呼び出し元のモジュールとしてのネットワークノード3において、機能識別子として機能識別コードが記憶されたアドレスを例示したが、機能識別子としては、文字などで示される機能名あるいは呼び出し元モジュールにおいてのみ有効な番号を用いれば、さらに高速な機能呼び出し処理を行うことができる。

【0064】また、ネットワークノード3、4相互間で入出力されるメッセージを蓄えるキューは、入力順に出力する他、所定の優先順位に基づいて、メッセージの出力順序を決定するようにしてもよい。すなわち、機能呼び出しの優先順位を付けるようにしてもよい。この優先順位は、予めシステムとして定められていてもよいし、メッセージに属性情報として含められていてもよい。

【0065】また、メッセージ送出側において、送出時刻をメッセージに含め、受信側において、その時刻に基づいて、そのメッセージで特定される機能の実行時刻を決定するようにしてもよい。また、メッセージを受信するキューは、各インスタンス（クラス）毎に存在してもよいし、複数のインスタンス（クラス）で共有するようにしてもよい。

【0066】また、本発明は、ネットワークノード3は、メッセージをネットワークノード4に出力した後に、ネットワークノード4からの処理結果を必要になるまで処理を続行し、必要となった時点で処理結果をまだ受け取っていない場合にのみ、実行中のプログラムの処理を停止するようにしてもよい。このとき、ネットワークノード3は、ネットワークノード4から、既に送ったメッセージについての処理結果を、設定時間を経過しても得られない場合に、自動的にエラー処理プログラムを起動するようにしてもよい。また、本発明は、ネットワ

ークノード3は、メッセージをネットワークノード4に出力した後に、ネットワークノード4からの処理結果を要求しないようにしてもよい。

【0067】

【発明の効果】以上説明したように、本発明の並列分散処理方法によれば、機能呼び出しを高速かつ効率的に行うことができる。また、本発明の並列分散処理方法によれば、ユーザによるモジュールの識別名の管理負担を軽減できる。また、本発明の並列分散処理方法によれば、機能呼び出しにおいて、多様な形式（データ構造）の引数を用いることができる。また、本発明の並列分散処理方法によれば、煩雑な作業を経ることなく、任意の呼び出し先のモジュールを動的にダウンロード（追加登録）して使用できる。

【図面の簡単な説明】

【図1】図1は、本発明の実施形態に係わる並列分散実行支援システム概念図である。

【図2】図2は、図1に示す並列分散実行支援システムにおいて、内部参照テーブルを作成する手法について説明するための図である。

【図3】図3は、図1に示す並列分散実行支援システムにおいて、外部参照テーブルを作成する手法について説明するための図である。

【図4】図4は、図1に示す並列分散実行支援システムにおいて、機能呼び出しを行う際の、呼び出し元のネットワークノードにおける処理を説明するための図である。

【図5】図5は、図1に示す並列分散実行支援システムにおいて、機能呼び出しを行う際の、呼び出し先のネットワークノードにおける処理を説明するための図である。

【図6】図6は、完全同期メッセージ送信を行う場合の図1に示す並列分散実行支援システムのタイミングチャートである。

【図7】図7は、非同期メッセージ送信における場合の図1に示す並列分散実行支援システムのタイミングチャートである。

【図8】図8は、本発明の並列分散実行支援システムの概念図である。

【符号の説明】

1…並列分散実行支援システム

2…ネットワーク

3、4…ネットワークノード

30、40…デモン

300、301、302、400、401、402…クラス

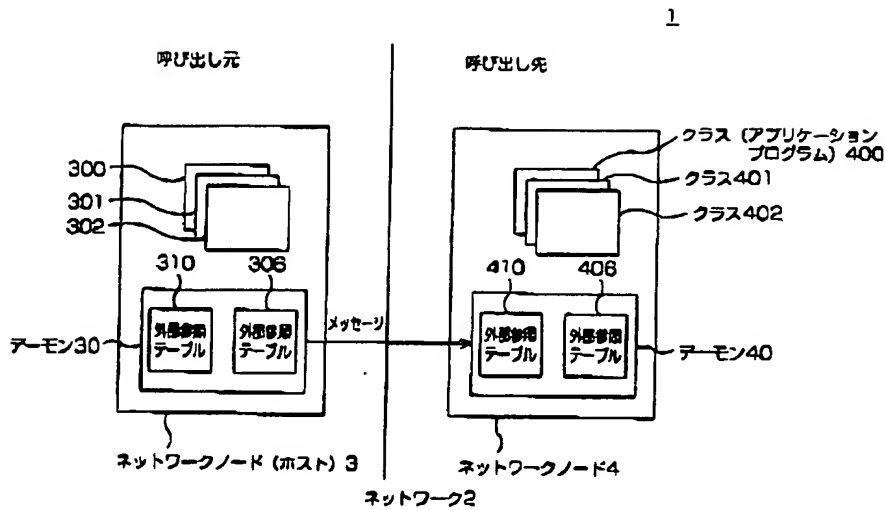
306、406…内部参照テーブル

310、410…外部参照テーブル

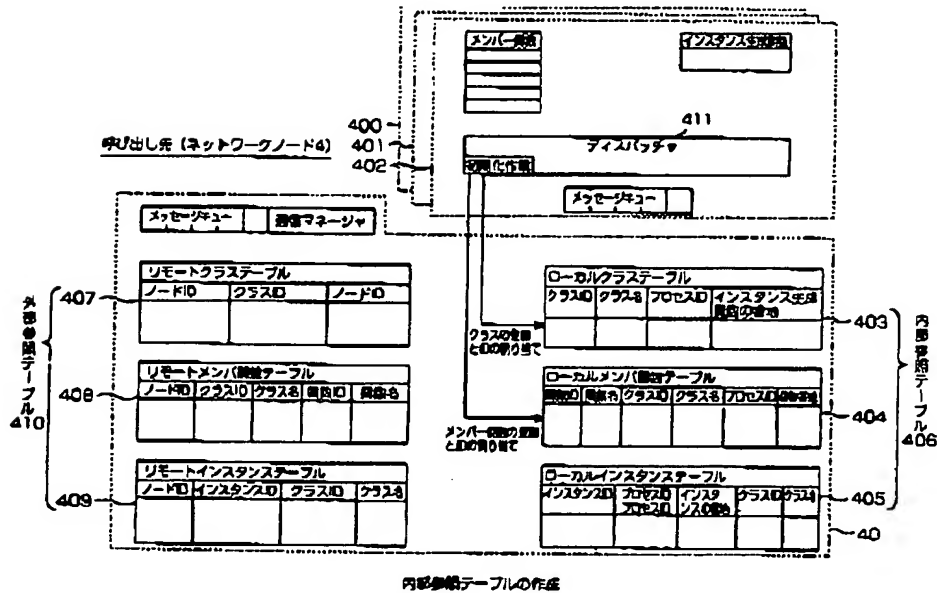
(10)

特開2002-82922

【图 1】



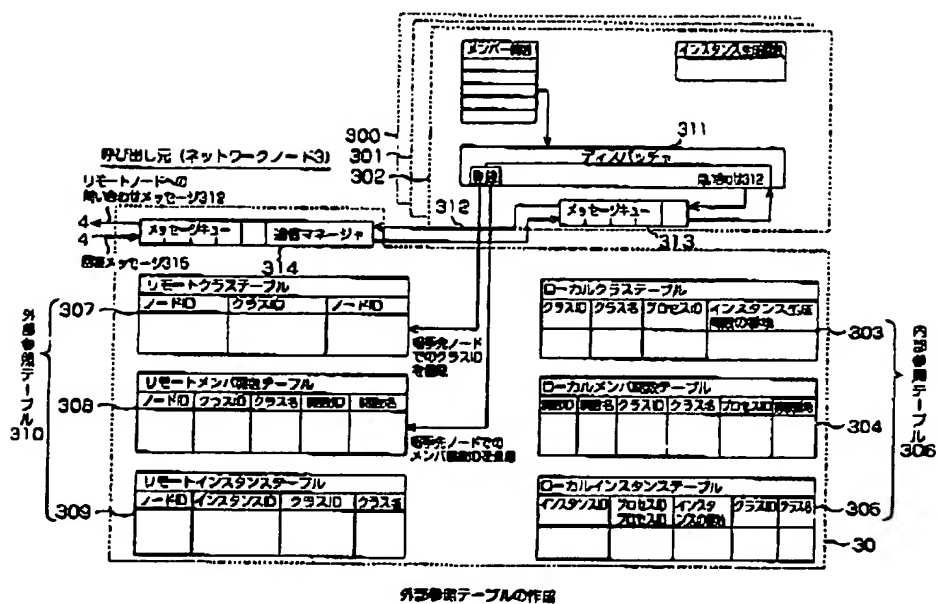
【图2】



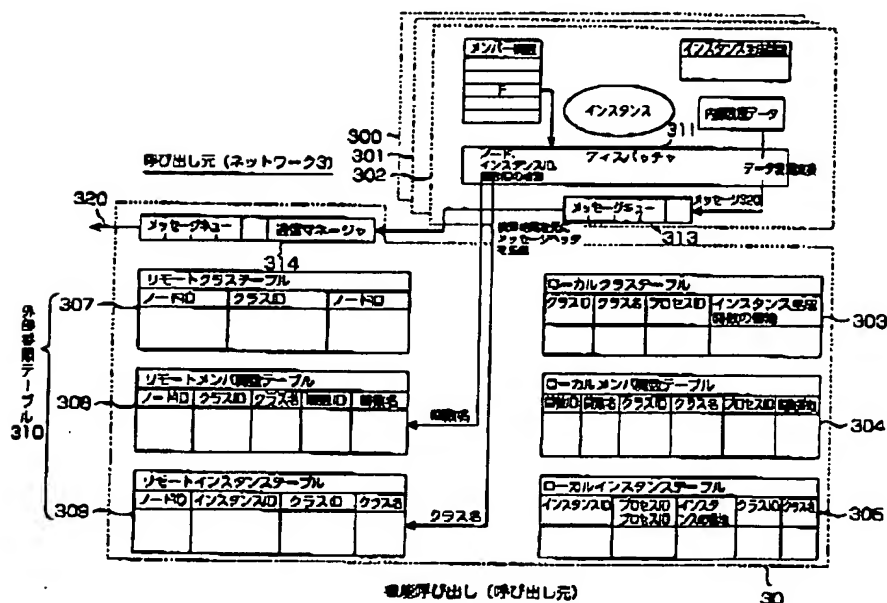
(11)

特開2002-82922

【図3】



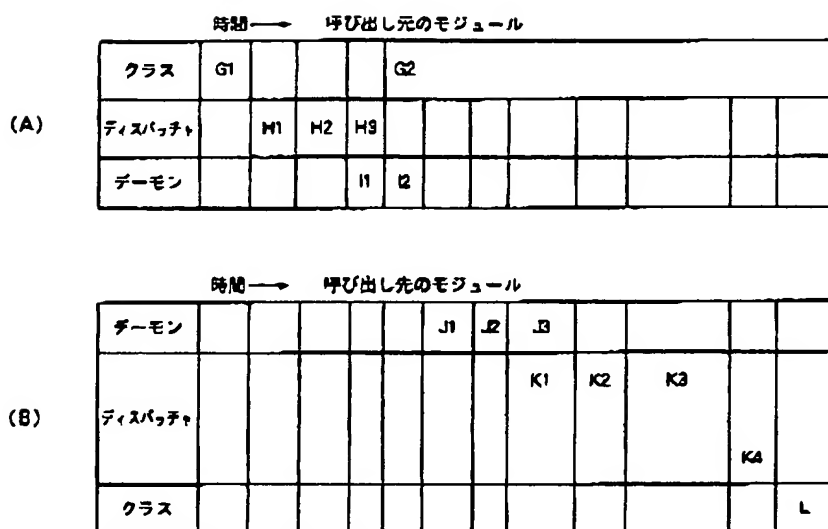
【図4】



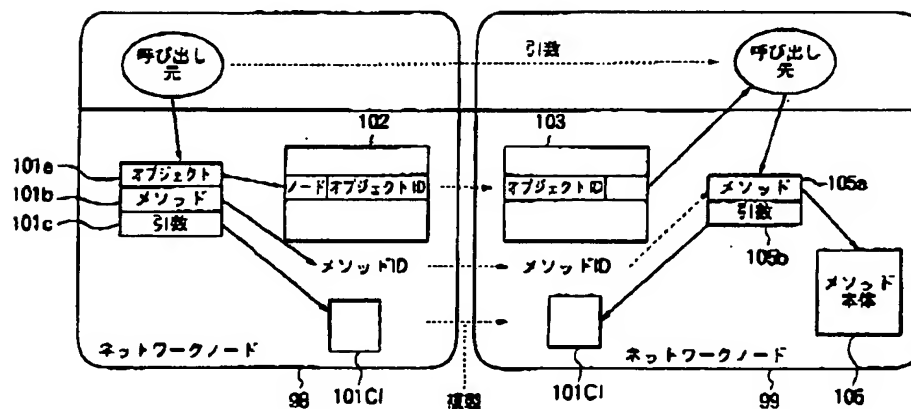
(13)

特開2002-82922

【図7】



【図8】



フロントページの続き

(72)発明者 千葉 哲央
 東京都港区赤坂七丁目3番37号 株式会社
 デジタル・ビジョン・ラボラトリーズ内

Fターム(参考) 5B045 GG01 GG11
 5B098 CA07 GC00